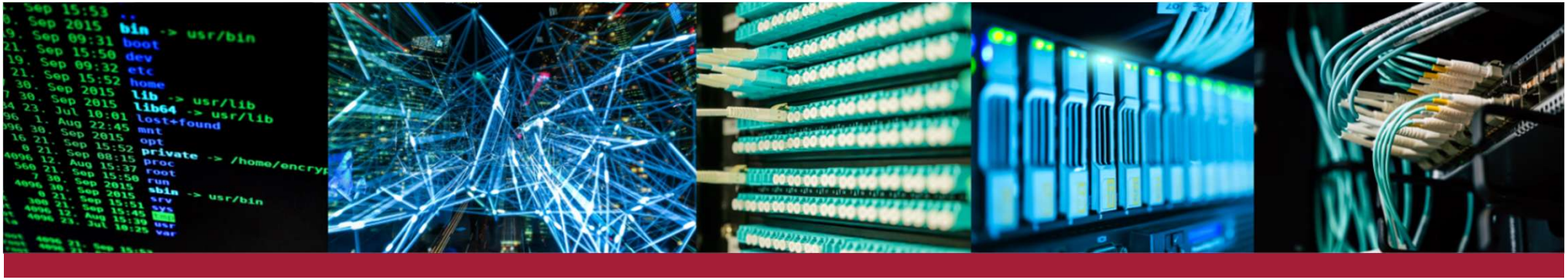
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Communication Networks



Implementation of Periodic Behavior with P4: Challenges and Solutions on Intel Tofino with Application in Time-Sensitive Networking (TSN)

Fabian Ihle <fabian.ihle@uni-tuebingen.de>

http://kn.inf.uni-tuebingen.de

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

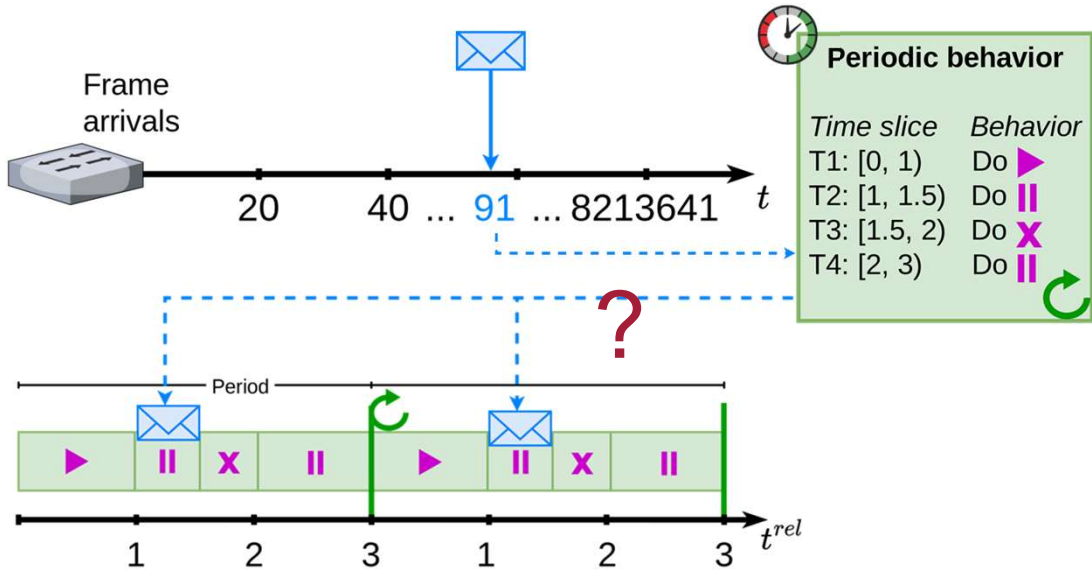
Agenda

- Motivation
- Background on TSN including PSFP and TAS, and the TNA
- Implementation of Periodic Behavior for PSFP
- Implementation of Periodic Queue Control for TAS
- Evaluation
- Conclusion

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Motivation

- Time-based, periodic mechanisms in switches
  - Apply behavior to frames based on a periodic schedule
    - Drop/forward, load balance, route, …
  - Use cases in TSN
- Hardware timestamps are available in P4 targets
  - Often an epoch timestamp
    - E.g., an absolute timestamp since the switch booted
    - Continuously increasing



- How to map the absolute timestamp to a periodically repeating schedule?
  - Solvable with arithmetics in theory
  - Not possible in practice due to constrained resources
    - → We need a trick to implement this

Time-Sensitive Networking, PSFP, and TAS

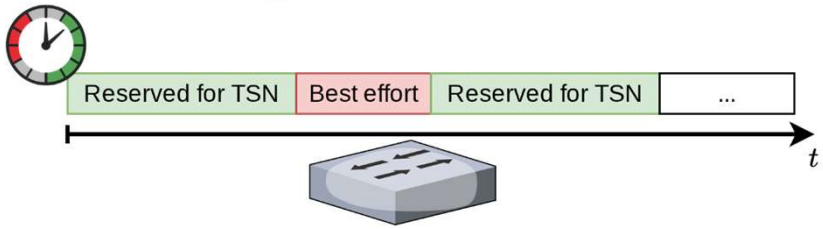
BACKGROUND

Time-Sensitive Networking (TSN)

- Time-Sensitive Networking (TSN) extends Ethernet with real-time components

- Scheduled traffic in TSN
  - High-priority real-time traffic (e.g., motion control) needs guaranteed low-jitter delivery
  - Sending times of talkers (senders) are coordinated to avoid congestion in intermediate nodes
    - Coordination is called scheduling and yields a network-wide schedule

- Capacity for scheduled traffic is reserved exclusively in intermediate bridges
  - → Frames traverse the network with bounded delay

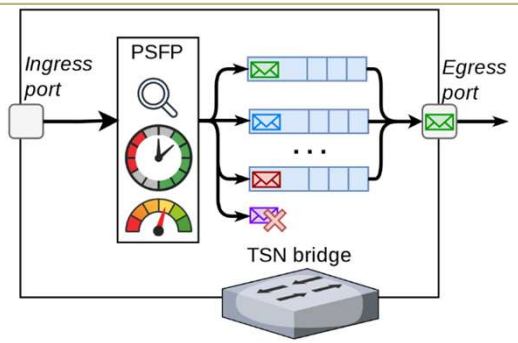


- Scheduled traffic (high priority) must be protected from low priority traffic (best effort)
  - Per-Stream Filtering and Policing (PSFP)
  - Time-Aware Shaper (TAS)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

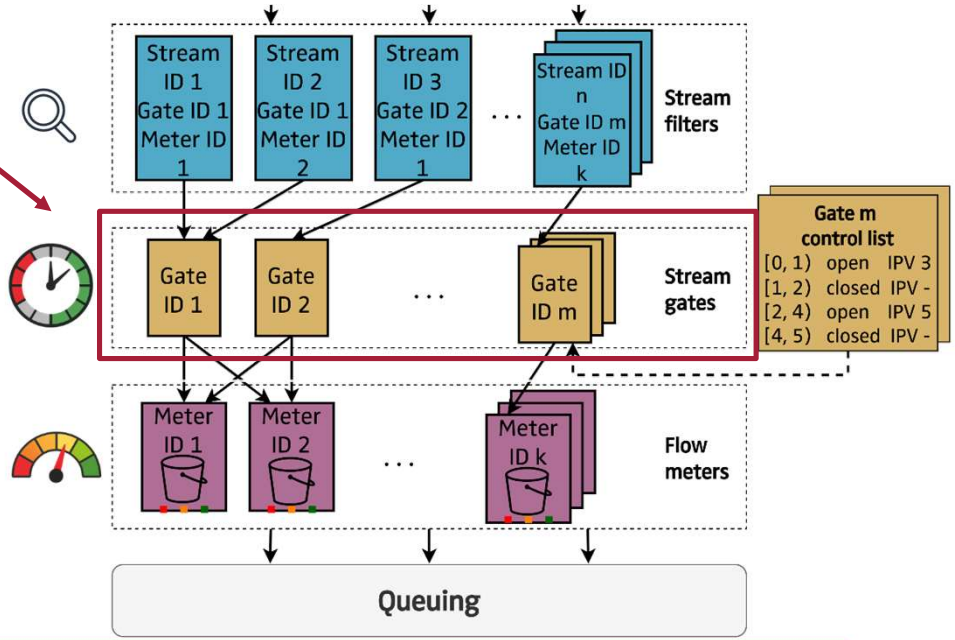
Per-Stream Filtering and Policing (PSFP)

- PSFP identifies and monitors streams regarding their reserved and allocated resources
  - Standardized in IEEE Std 802.1Qci
  - Implemented in the **ingress** of TSN bridges



We focus on this

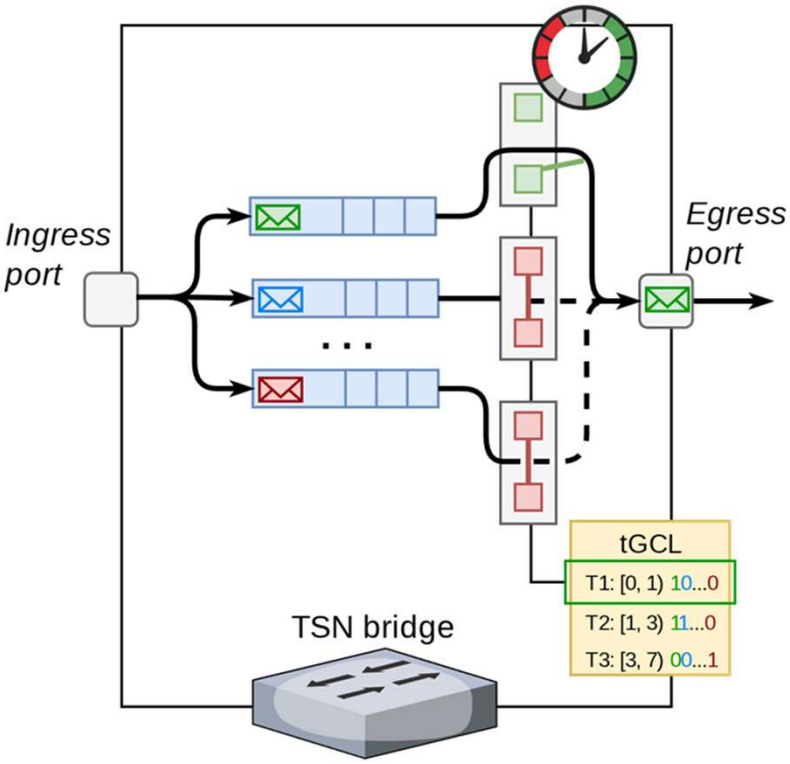
- PSFP applies…
  - Time-based metering with stream gates
    - Admitted transmission times are contained in a stream Gate Control List (**sGCL**)
      - **Periodically** repeated time slices with open / closed state
      - Forwards / drops frames according to sGCL
  - Credit-based metering
    - Forwards / drops frames according to rate limit



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

The Time-Aware Shaper (TAS)

- TSN frames are queued into one of eight FIFO queues
  - Based on frame priority in VLAN tag
- The TAS protects high-priority traffic from low-priority traffic by controlling priority queues
  - Standardized in IEEE Std 802.1Qbv
  - Implemented in the **egress** of TSN bridges
- Priority queues are opened / closed based on a **periodic** time-based GCL
  - The transmission GCL (**tGCL**)
    - Periodically repeated time slices with 8-bit vector
      - Each bit controls one of the eight queues

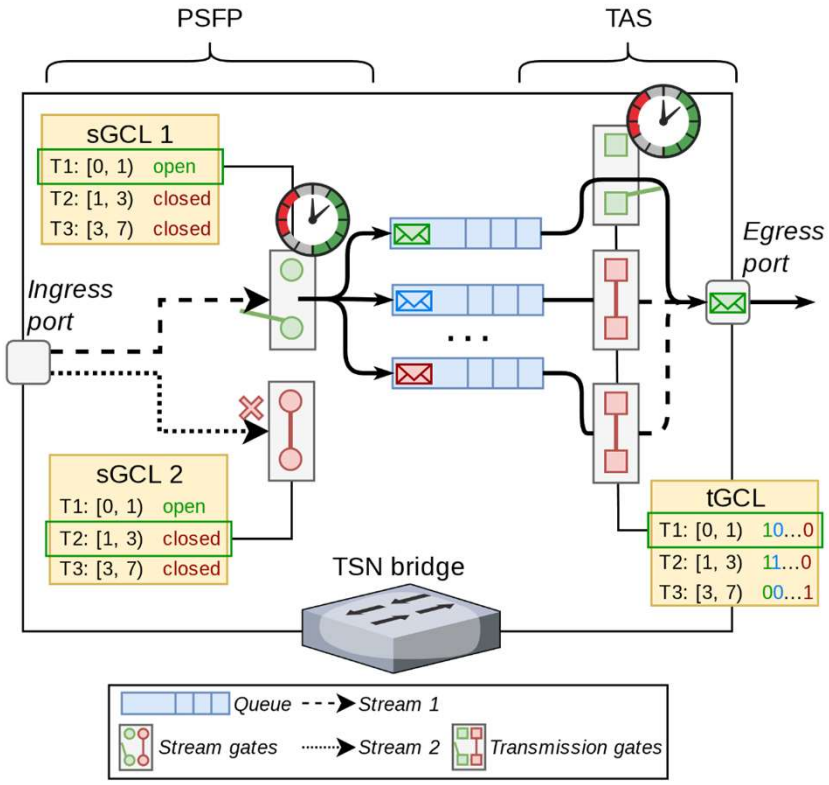


EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Summary – Time-based Metering with TAS and PSFP

- PSFP
  - Ingress policing
    - Applied per stream
    - Action: drop frames or overwrite their priority
  - Time-based stream gates controlled by **periodic** sGCL

- TAS
  - Egress shaping
    - Applied per egress port
    - Action: forward or buffer frames by opening / closing priority queues
  - Time-based transmission gates controlled by **periodic** tGCL



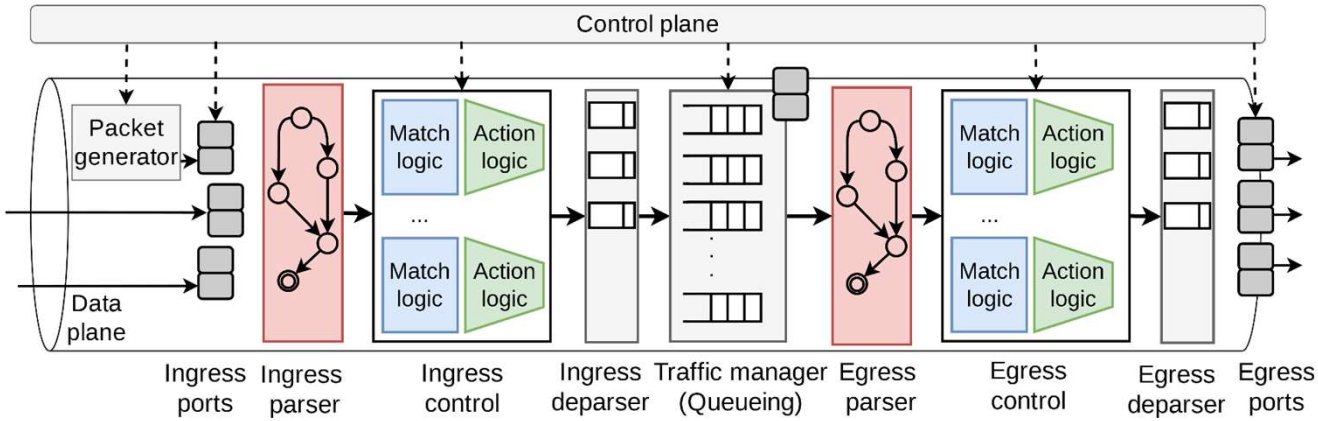
→ TAS and PSFP in P4: **periodic** behavior required for GCLs

The Intel Tofino Switching ASIC

- P4 programmable switching ASIC
  - Programmable ingress/egress parser, control block, deparser
  - 32x 100/400 Gb/s
  - Each egress port has multiple hardware queues

- Hardware timestamping
  - Nanosecond resolution (48 bit field)
  - Precision Time Protocol (PTP) integration possible

- Internal packet generator
  - Dedicated packet generation port
  - Packet generation configurable
    - Burst size
      - Generate B batches with K packets per batch
    - Periodic trigger
      - Generate packets every x ns



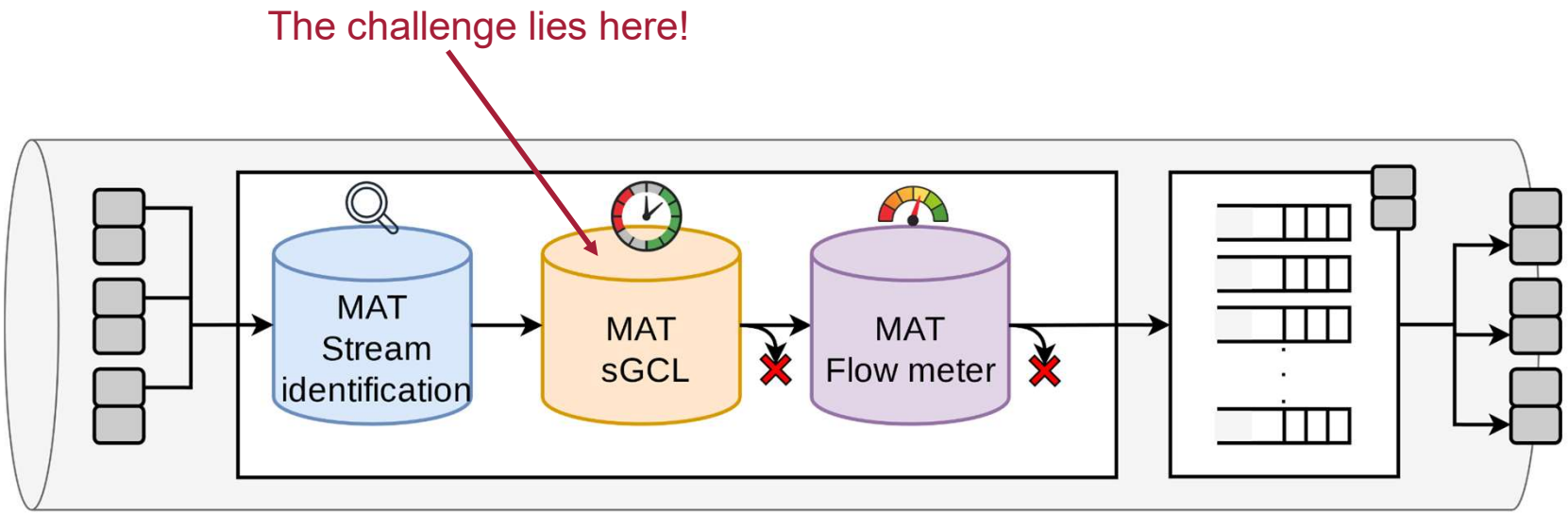
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Implementation Concept

IMPLEMENTATION OF PERIODIC BEHAVIOR WITH P4 FOR PSFP

P4-PSFP Pipeline Overview

- All components of the PSFP mechanism are implemented using MATs, registers and TNA externs
  → Basic P4 building blocks enable sophisticated mechanisms such as PSFP

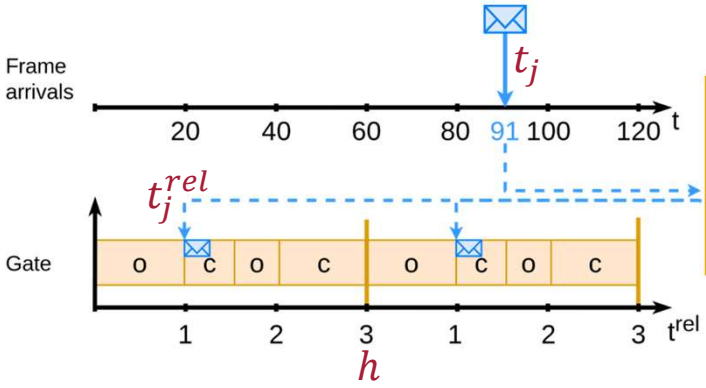


EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Challenge – Time-based Metering

- How to implement a time-based schedule in P4?
  - Packets have an ingress hardware timestamp t_j ☺
- Model the GCL as a MAT ☺
  - Contains the GCL time slices as entries
  - Matches on the ingress timestamp
- How to model periodicity of GCLs? ☹
  - How to convert hardware timestamp t_j to a relative position t_j^{rel} in the GCL with period h?
    - $t_j^{rel} = (t_j - t_0) \; mod \; h$

No modulo operator available on Tofino!



Stream gate control list

Time slice	*State*	*Priority*
[0, 1)	open	IPV 3
[1, 1.5)	closed	IPV -
[1.5, 2)	open	IPV *
[2, 3)	closed	IPV -

MAT_sGCL

Match keys:
stream_gate_id: exact &
relative_timestamp: range

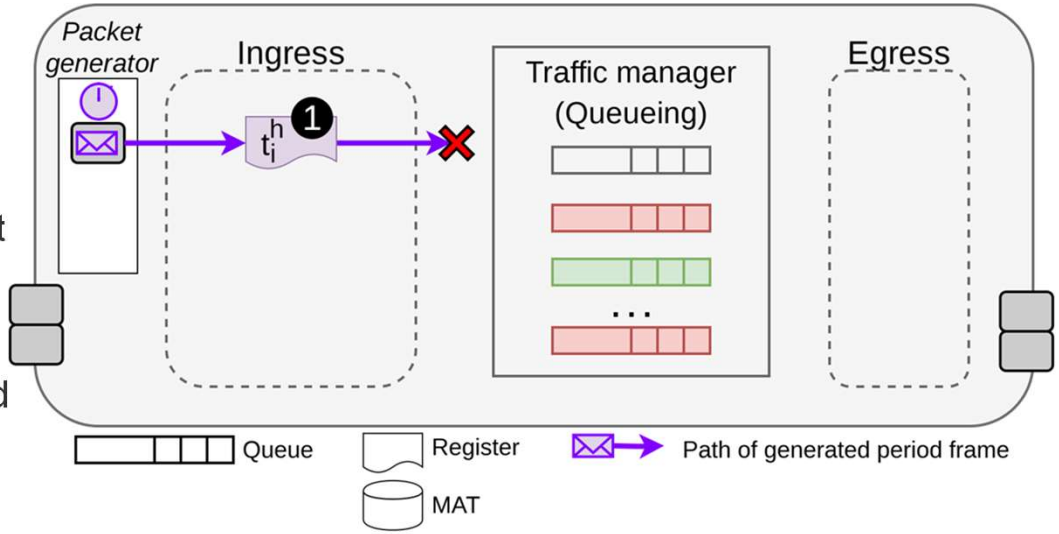
	Match fields		Action
	stream_gate_id	rel_ts	
sGCL 1	1	[0, 1)	*modify_queue(prio=3)*
	...	[1, 1.5)	*drop()*
	1	[1.5, 2)	*forward()*
	1	[2, 3)	*drop()*
sGCL 2	2	[1, 3)	...
	...	[3, 5)	...
	2	[0, 4)	...

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Solution – Periodicity of GCLs

❶ Configure the internal packet generator to generate *period-completion frames*

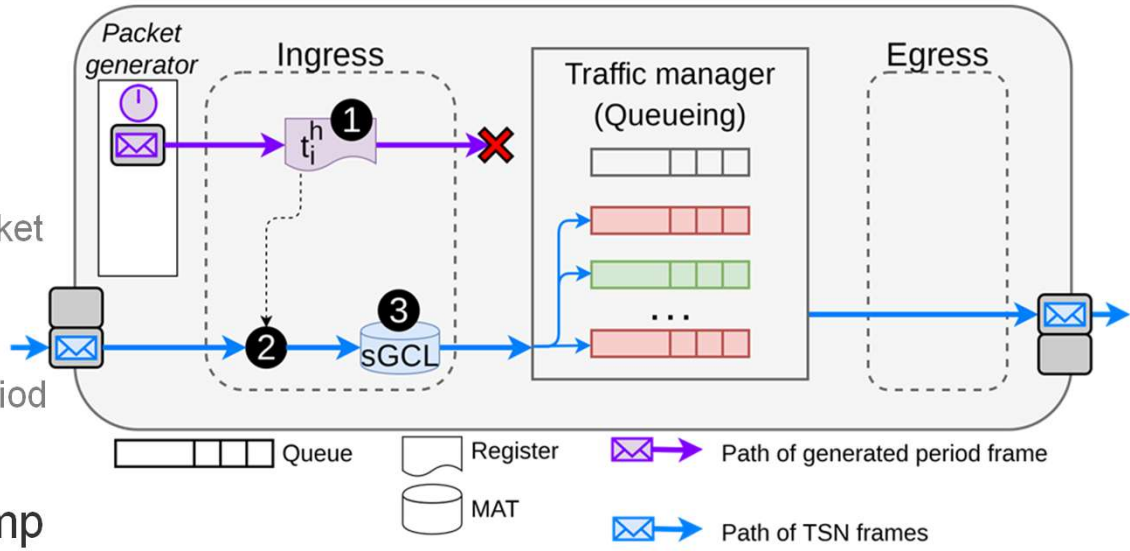
1. Generate a frame i every h ns
   - Frame is processed in the pipeline via internal packet generation port
2. Store ingress timestamp t_i^h in a register
   - t_i^h reflects the timestamp of the last completed period



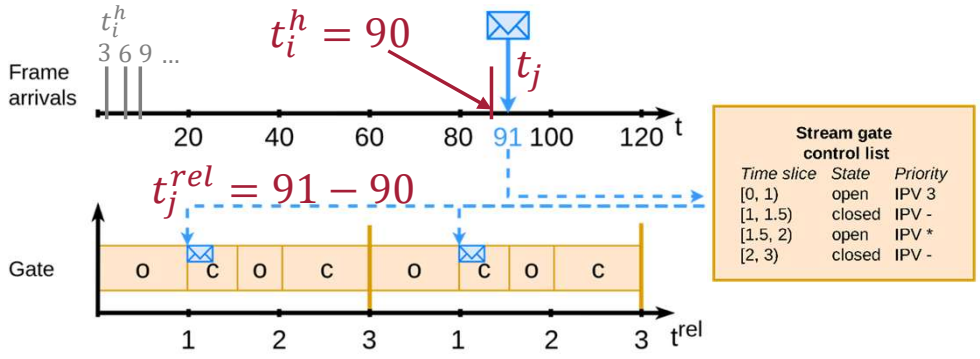
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Solution – Periodicity of GCLs

1. Configure the internal packet generator to generate *period-completion frames*
   1. Generate a frame i every h ns
      - Frame is processed in the pipeline via internal packet generation port
   2. Store ingress timestamp t_i^h in a register
      - t_i^h reflects the timestamp of the last completed period



2. For each TSN frame j with absolute timestamp t_j ...
   - Calculate relative position t_j^{rel} in GCL based on t_i^h
     - $t_j^{rel} = t_j - t_i^h$ → Semantically a modulo operation

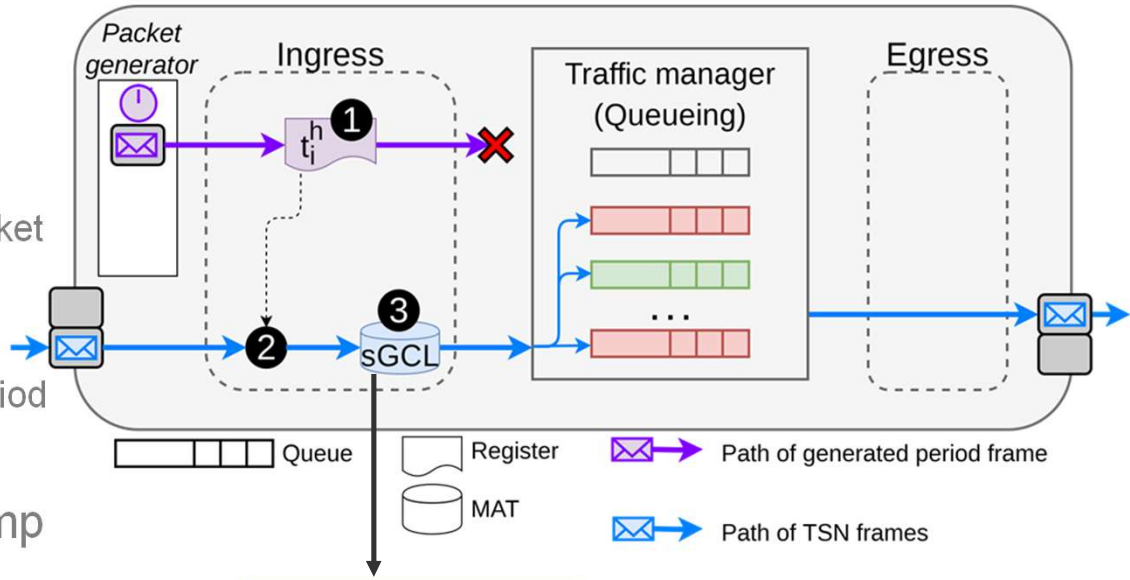


Solution – Periodicity of GCLs

1. Configure the internal packet generator to generate *period-completion frames*
   1. Generate a frame i every h ns
      - Frame is processed in the pipeline via internal packet generation port
   2. Store ingress timestamp t_i^h in a register
      - t_i^h reflects the timestamp of the last completed period

2. For each TSN frame j with absolute timestamp t_j ...
   - Calculate relative position t_j^{rel} in GCL based on t_i^h
     - $t_j^{rel} = t_j - t_i^h$ → Semantically a modulo operation

3. Apply PSFP
   - Match in sGCL based on t_j^{rel}



MAT_sGCL

Match keys:
stream_gate_id: exact &
relative_timestamp: range

	Match fields		Action
	stream_gate_id	rel_ts	
sGCL 1	1	[0, 1)	*modify_queue(prio=3)*
	...	[1, 1.5)	*drop()*
	1	[1.5, 2)	*forward()*
	1	[2, 3)	*drop()*
sGCL 2	2	[1, 3)	...
	...	[3, 5)	...
	2	[0, 4)	...

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Summary – Time-based Metering for TSN in P4

- TAS and PSFP are mechanisms to protect scheduled traffic
  - Both use periodic Gate Control Lists (GCLs) for time-based metering
- Periodicity of GCLs in P4 is challenging
  - Limited arithmetics and resources in hardware
- Use the internal packet generator of the Intel Tofino as a clock source
  1. Generate one packet after every period
  2. Store timestamp of generated packet
  - → Timestamp references last completed period

MAT_sGCL

Match keys:
stream_gate_id: exact &
relative_timestamp: range

	Match fields		Action
	stream_gate_id	rel_ts	
sGCL 1	1	[0, 1)	*modify_queue(prio=3)*
	...	[1, 1.5)	*drop()*
	1	[1.5, 2)	*forward()*
	1	[2, 3)	*drop()*
sGCL 2	2	[1, 3)	...
	...	[3, 5)	...
	2	[0, 4)	...

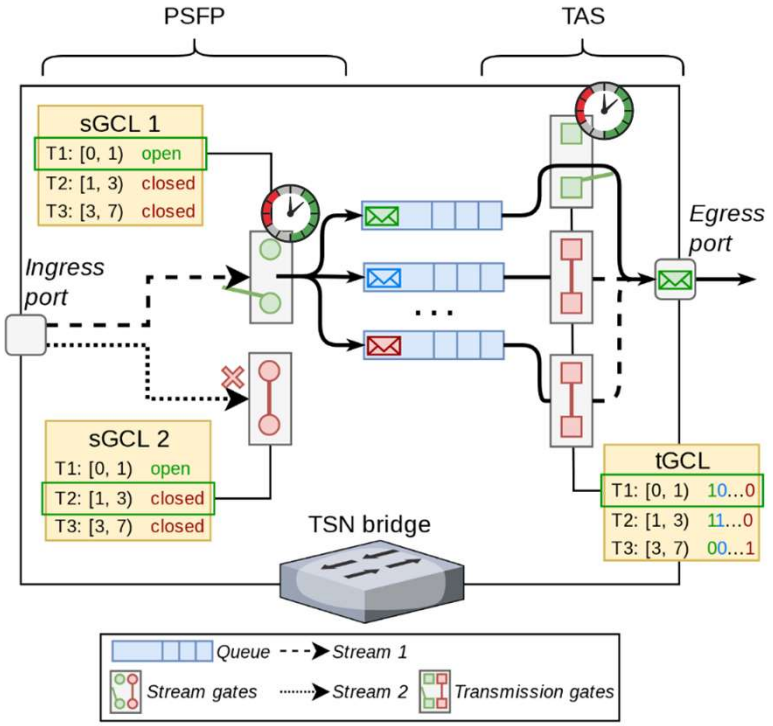
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Implementation Concept

IMPLEMENTATION OF PERIODIC QUEUE CONTROL WITH P4 FOR TAS

Periodic Queue Control for TAS

- The TAS and PSFP both use periodic GCLs
  - We can use the mechanism for periodic behavior with P4 for PSFP and TAS
- The action in PSFP is simple
  - Drop, forward, change priority
- For TAS, the action is more complex
  - We need to periodically control queue states!
    - How to do this?

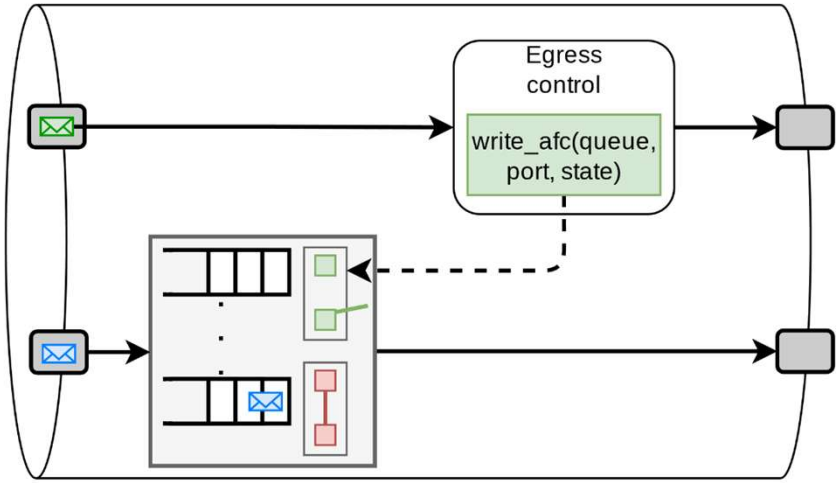


How to Control Queue States?

- The Tofino 2 introduces AdvancedFlowControl (AFC)
  - Allows to open / close a queue „manually“
  - Triggered by a frame during pipeline processing
    - Writing a value into the AFC metadata field

- The AFC metadata field contains…
  - the egress port number
  - the queue ID
  - the state (open/close)

- AFC is not limited to the frame’s own port / queue!
  - Can control arbitrary queues of other ports



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Challenge – Continuous Queue Control

- The tGCL follows the same concept as the sGCL ☺
  - Time slices as entries
  - Reuse mechanism for periodic behavior

- Key challenges
  - TAS controls egress queue availability over time
    - For each tGCL entry, we have to control 8 queue states
    - **One** frame can only control **one** queue ☹
  - Queue state changes are not instant
    - If TSN frames update their own queue state, queue states are updated too late
    - Queue openings must occur before packets arrive ☹

tGCL	
T1: [0, 1)	10…0
T2: [1, 3)	11…0
T3: [3, 7)	00…1

MAT_tGCL

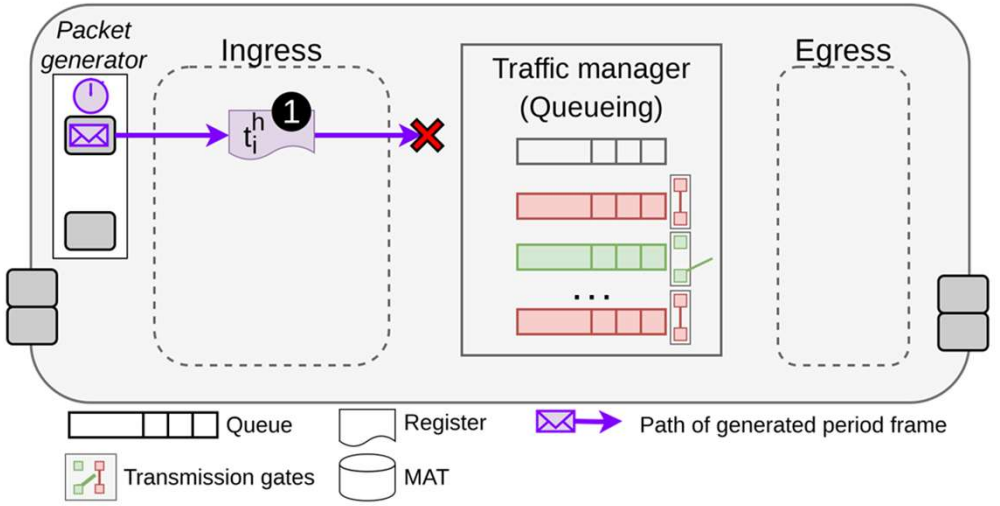
Match keys:
queue_id: exact &
egress_port exact &
relative_timestamp: range

	Match fields			Action	
	queue_id	egress_port	rel_ts		
tGCL port X	0	X	[0, 1)	*open_queue(id=0, port=X)*	First time slice of tGCL port X
	…	X	[0, 1)	…	
	7	X	[0, 1)	*close_queue(id=7, port=X)*	
	0	X	[1, 3)	*close_queue(id=0, port=X)*	
	…	X	[1, 3)	…	
	7	X	[1, 3)	*open_queue(id=7, port=X)*	
tGCL port Y	0	Y	[0, 4)	*open_queue(id=0, port=Y)*	
	…	Y	[0, 4)	…	

- We need a timely mechanism that continuously updates all queue states according to the tGCL
  - → Use the internal traffic generator to generate a continuous TAS control stream

The P4-TAS Mechanism (1)

❶ Generate a period-completion frame every h ns and store its ingress timestamp t_i^h in a register

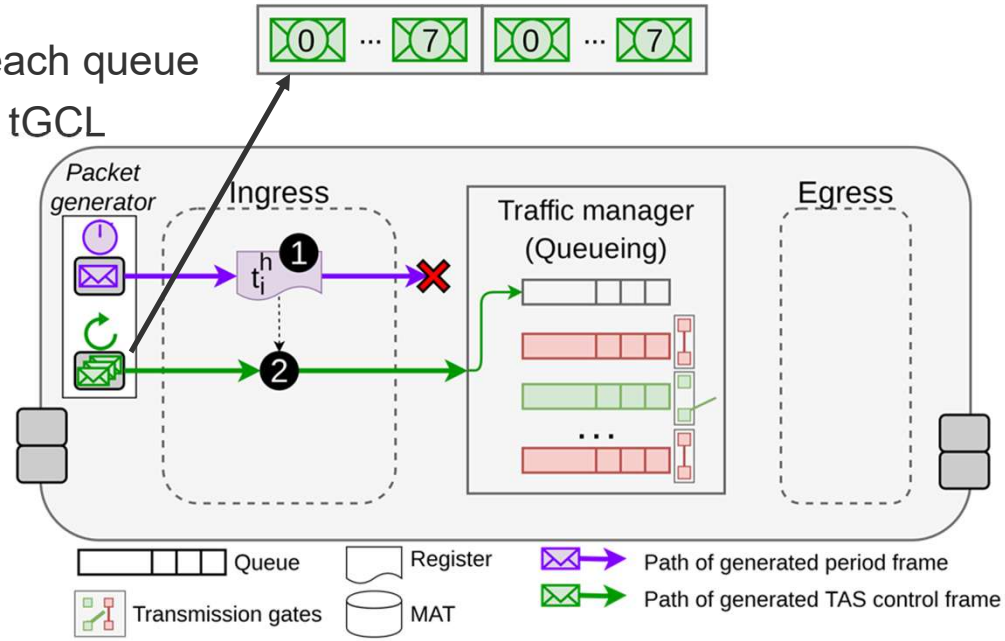


The P4-TAS Mechanism (2)

1 Generate a period-completion frame every h ns and store its ingress timestamp t_i^h in a register

2 Continuously generate frames k with a minimum IAT

- = TAS control traffic
- Packet bursts of eight at a time, one for each queue
- Calculate their relative position t_k^{rel} in the tGCL
- Queue into a dedicated egress queue

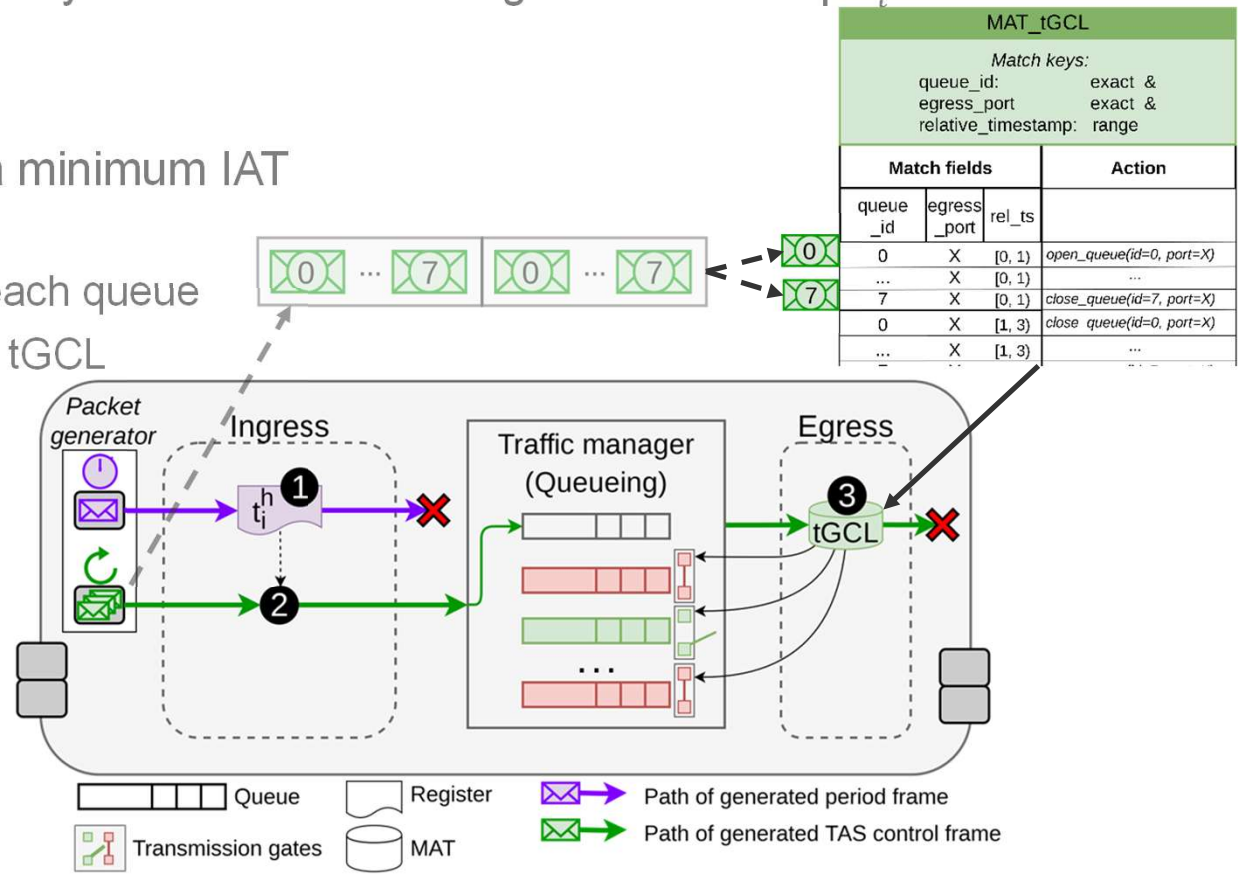


The P4-TAS Mechanism (3)

1. Generate a period-completion frame every h ns and store its ingress timestamp t_i^h in a register

2. Continuously generate frames k with a minimum IAT
   - = TAS control traffic
   - Packet bursts of eight at a time, one for each queue
   - Calculate their relative position t_k^{rel} in the tGCL
   - Queue into a dedicated egress queue

3. Match TAS control traffic based on relative position t_k^{rel} in tGCL MAT
   - Open / close queues of egress port
   - Drop frame



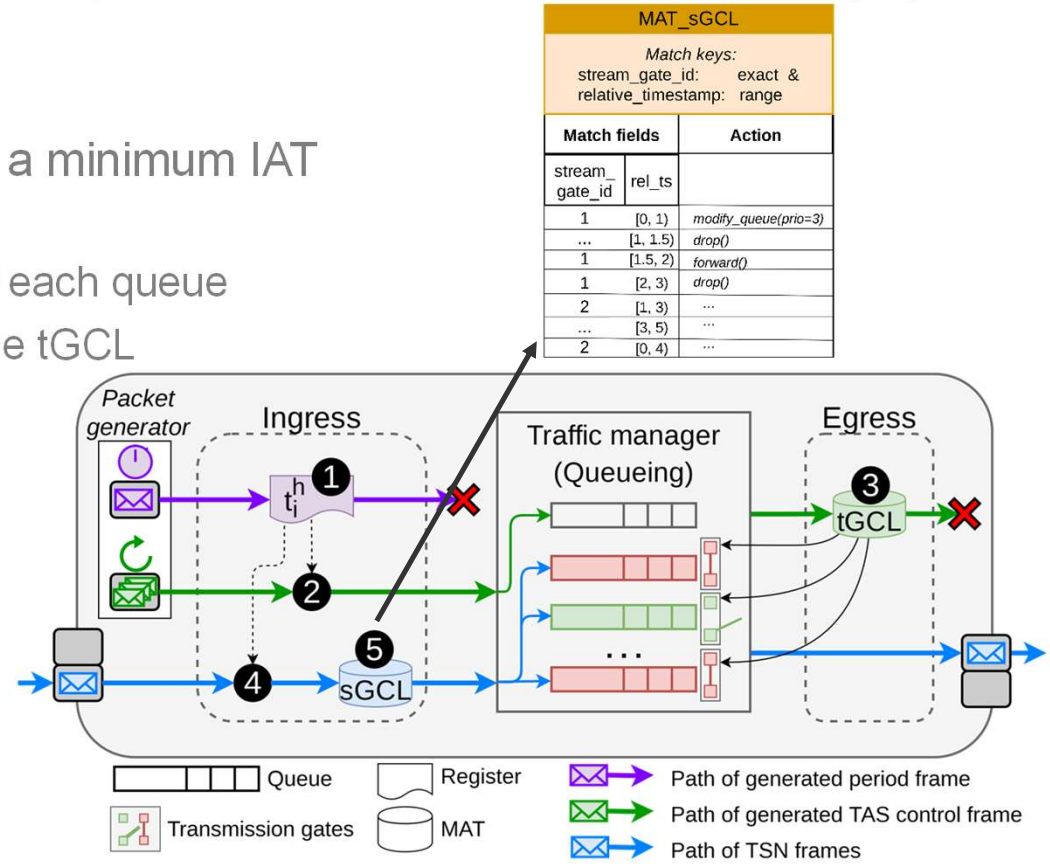
The P4-TAS Mechanism (4)

1. Generate a period-completion frame every h ns and store its ingress timestamp t_i^h in a register

2. Continuously generate frames k with a minimum IAT
   - = TAS control traffic
   - Packet bursts of eight at a time, one for each queue
   - Calculate their relative position t_k^{rel} in the tGCL
   - Queue into a dedicated egress queue

3. Match TAS control traffic based on relative position t_k^{rel} in tGCL MAT
   - Open / close queues of egress port
   - Drop frame

4. For TSN traffic, calculate relative position t_i^{rel} in sGCL, 5. apply PSFP



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Summary – Periodic Queue Control with P4

- The mechanism for periodic behavior can be reused for the TAS
  - The action is more complex compared to PSFP
    - Periodically control multiple queue states
- The Tofino 2 enables „manual“ opening and closing of queues

MAT_tGCL

Match keys:
queue_id: exact &
egress_port exact &
relative_timestamp: range

	Match fields			Action	
	queue_id	egress_port	rel_ts		
tGCL port X	0	X	[0, 1)	*open_queue(id=0, port=X)*	First time slice of tGCL port X
	...	X	[0, 1)	...	
	7	X	[0, 1)	*close_queue(id=7, port=X)*	
	0	X	[1, 3)	*close_queue(id=0, port=X)*	
	...	X	[1, 3)	...	
	7	X	[1, 3)	*open_queue(id=7, port=X)*	
tGCL port Y	0	Y	[0, 4)	*open_queue(id=0, port=Y)*	
	...	Y	[0, 4)	...	

- Challenges
  - A single frame can only control a single queue, but we need to control 8 queues
  - Queue state changes are not instant → TSN traffic cannot control its own queue
- Use the internal packet generator of the Intel Tofino for continuous TAS control traffic
  1. Generate a continuous stream with packet bursts of 8
  2. Calculate relative position in tGCL for those packets
  3. Match packets in tGCL MAT → open/close corresponding queue

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

EVALUATION

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Evaluation – Traffic Generator Accuracy

- The internal packet generator is used for generating…
  a) Period-completion frames
     - Generate a frame every h ns to indicate the end of a GCL period
  b) Continuous TAS control traffic
     - Generate 8 frames every 1 ns to continuously poll for queue state changes

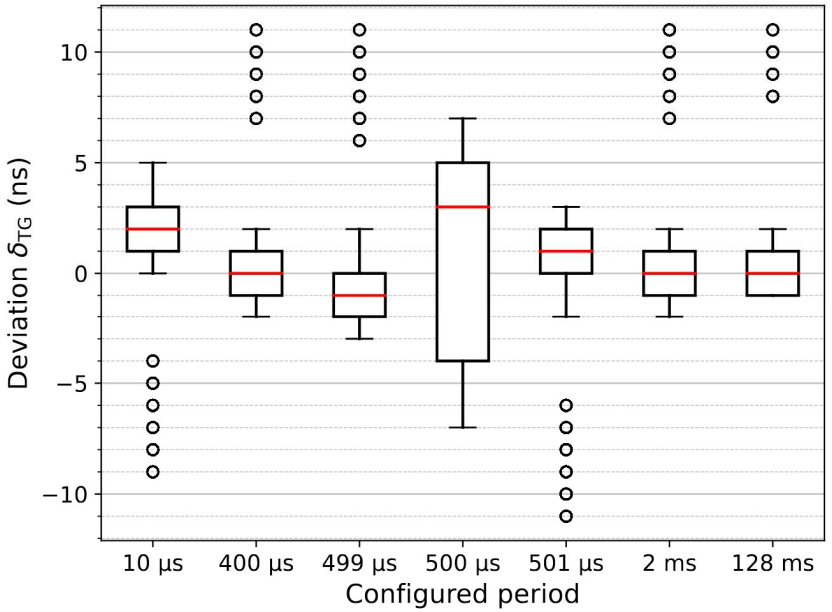
- The packet generator is not 100% accurate
  a) Period-completion frames may arrive early / late
  b) The minimum IAT of 1 ns is practially not possible

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Evaluation – Traffic Generator Accuracy

a) Problem: period-completion frames may arrive early / late

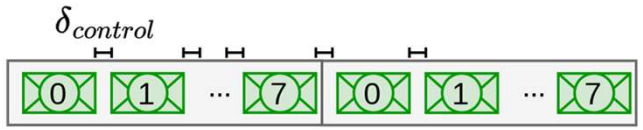
- Measurement
  - Configure different periods $h \in \{10\ \mu s, 400\ \mu s, 500\ \mu s, 2\ ms, 128\ ms\}$
  - Collect timestamps t_i^h of *i-th* period with duration h in a data plane register
  - Calculate deviation $\delta_{TG} = \left(t_{i+1}^h - t_i^h\right) - h$
- Interpretation
  - $\delta_{TG} = 0$ → perfect
  - $\delta_{TG} > 0$ → period was too long
  - $\delta_{TG} < 0$ → period was too short
- Result
  - The deviation is very small $(\pm 11\ ns)$
  - Can impact period computation



Evaluation – Traffic Generator Accuracy

b) Problem: a minimum IAT of 1 ns is practically not possible

- Used by TAS control traffic for timely queue control

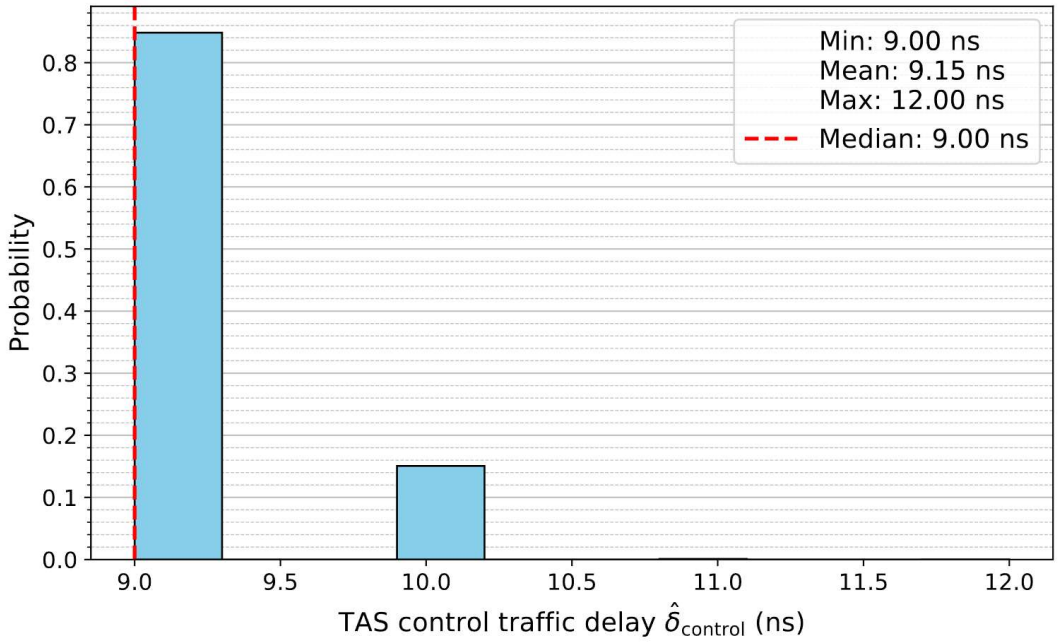


▶ Measurement

- Track delay $\delta_{control}$ between generated TAS control frames

▶ Result

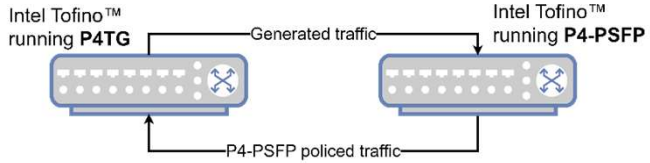
- One frame can be generated every 9 ns
  - → Queue updates can be only triggered every 9 ns
  - Frames in a batch are generated sequentially
  - → Not all queues are updated simultaneuosly!



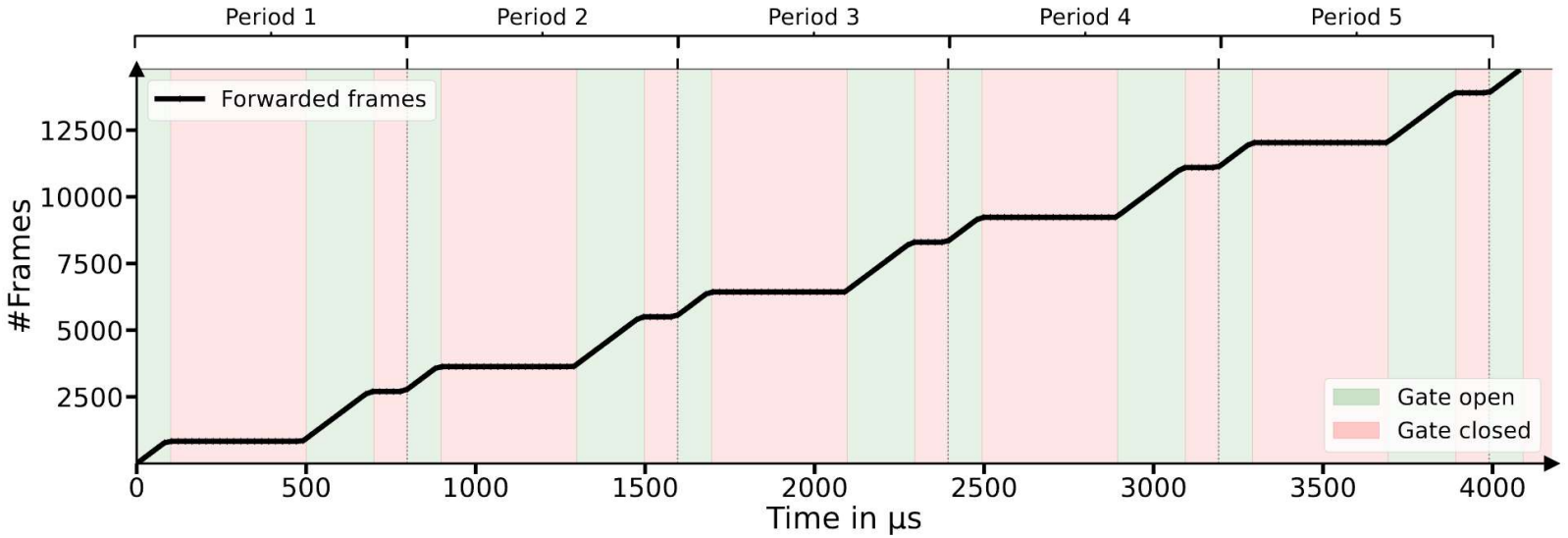
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Evaluation – Time-based Metering and Periodicity

- Validate the periodicity mechanism
  - Apply time-based metering: 1-4-2-1 stream GCL
    - 100 µs open, 400 µs closed, 200 µs open, 100 µs closed
  - Measure #frames over time



- Traffic generator P4TG [3] generates a 100 Gb/s stream



[2] S. Lindner, M. Häberle, and M. Menth, “P4TG: 1Tb/s Traffic Generation for Ethernet/IP Networks”, IEEE Access, 2023.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

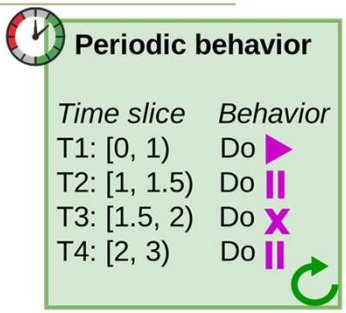
Conclusion

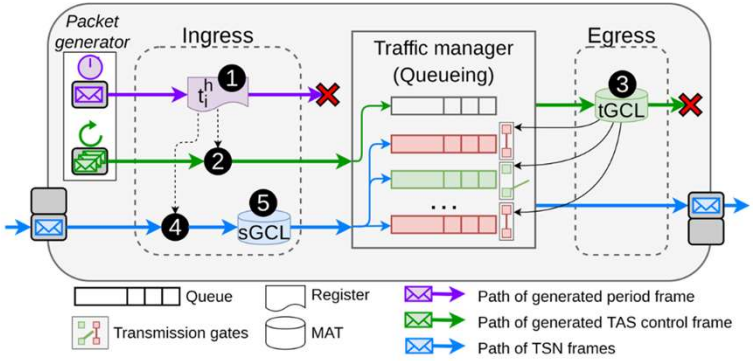
- Periodic behavior in P4 switches is challenging
  - Limited arithmetics and resources in hardware
  - Required for TSN mechanisms such as PSFP and TAS

- We presented a mechanism…
  a) to achieve periodic MATs in P4
     – Using the internal traffic generator of the Intel Tofino as a clock source
  b) to achieve periodic queue control
     – Generation of continuous TAS control traffic

- Open source
  - https://github.com/uni-tue-kn/P4-PSFP
  - https://github.com/uni-tue-kn/P4-TAS

- Publications
  - P4-PSFP: https://ieeexplore.ieee.org/document/10611744 (preprint)
  - P4-TAS: http://arxiv.org/abs/2511.10249 (under submission)





EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Questions?

THANK YOU!

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

References

[1]: F. Ihle, S. Lindner and M. Menth: P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking, in IEEE TNSM, Special Issue on Robust and Resilient Future Communication Networks, vol. 21, no. 5, p. 5273 - 5290, Oct. 2024, IEEE

[2]: P. Gupta and N. McKeown, “Algorithms for Packet Classification,” IEEE Network, vol. 15, no. 2, pp. 24–32, 2001.

[3]: S. Lindner, M. Häberle, and M. Menth, “P4TG: 1Tb/s Traffic Generation for Ethernet/IP Networks”, IEEE Access, 2023.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

APPENDIX

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Challenge – Timestamp Matching Precision

- Entries in the GCL are time slices (intervals)
  - P4 supports the range matching type ☺
  - We have 48 bit timestamps (in ns) ☺
  - But our target is limited ☹
    - Can only range match on max. 20 bit

- 20 bit range matching allows for a range of [1 ns, ~1 ms]
  - With bit shifting, we can achieve [2 µs, ~4 s]
  - → We want more!
    - Nanosecond precision and large periods

- → Employ range-to-ternary conversion algorithm to achieve a higher matching resolution

MAT_sGCL

Match keys:
stream_gate_id: exact &
relative_timestamp: range

	Match fields		Action
	stream_gate_id	rel_ts	
sGCL 1	1	[0, 1)	*modify_queue(prio=3)*
	...	[1, 1.5)	*drop()*
	1	[1.5, 2)	*forward()*
	1	[2, 3)	*drop()*
sGCL 2	2	[1, 3)	...
	...	[3, 5)	...
	2	[0, 4)	...

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Range-to-Ternary Conversion

- Convert a single range matching entry to multiple wildcard ternary entries
  - Iteratively select the largest prefix starting at the current lower bound that fits into the range
  - Ternary entries collectively cover the entire range

- Tradeoff
  - \+ We can use the entire timestamp range
  - – We need much more MAT space
    - Range of bit width w requires at most $2w - 2$ entries [2]
    - → We have enough MAT space to do this ☺